A Method for Obtaining Highly Robust Memristor Based Binarized Convolutional Neural Network

Lixing Huang[1], Jietao Diao[1], Shuhua Teng[2], Zhiwei Li[1], Wei Wang[1], Sen Liu[1], Minghou Li[3], and Haijun Liu[1(✉)]

[1] College of Electronic Science and Technology, National University of Defence Technology, Changsha 410073, Hunan, China
liuhaijun@nudt.edu.cn

[2] Hunan Communications Research Institute Co., Ltd., Changsha 410015, Hunan, China

[3] Qingdao Geo-Engineering Surverying Institute, Qingdao 266100, Shandong, China

Abstract. Recently, memristor based binarized convolutional neural network has been widely investigated owing to its strong processing capability, low power consumption and high computing efficiency.However, it has not been widely applied in the field of embedded neuromorphic computing for manufacturing technology of the memristor being not mature. With respect to this, we propose a method for obtaining highly robust memristor based binarized convolutional neural network. To demonstrate the performance of the method, a convolutional neural network architecture with two layers is used for simulation, and the simulation results show that binarized convolutional neural network can still achieve more than 96.75% recognition rate on MNIST dataset under the condition of 80% yield of the memristor array, and the recognition rate is 94.53% when the variation of memristance is 26%, and it is 94.66% when the variation of the neuron output is 0.8.

Keywords: Memristor · Binarized convolutional neural network · Variation

1 Introduction

Binarized convolutional neural network [1, 2] has obtained much attention owing to its excellent computing efficiency [3] and fewer storage consumption [4]. However, when faced with complex tasks [5], the depth of the neural network will become deeper and deeper [6], increasing the demands on the communication bandwidth. And constrained by the problem of memory wall [7] in von Neumann architecture, it is difficult to realize further improvement in computing speed and energy efficiency.

Fortunately, the emerging of memristor [8] based computing system provides a novel processing architecture, viz., processing-in-memory (PIM) architecture [9], solving the memory wall problem existed in von Neumann architecture. Because the core computing component in PIM architecture, memristor array, is not only used to store weights of neural network but also to execute matrix-vector multiplier, data transferring between memory and computing units is avoided, thus decreasing the requirements of communication bandwidth and improving computing speed and energy efficiency.

Z. Qian et al. (Eds.): WCNA 2021, LNEE 942, pp. 813–822, 2022.
https://doi.org/10.1007/978-981-19-2456-9_82

Nevertheless, the manufacturing technology of the memristor is still not mature, the manufactured devices existing many non-ideal characteristics [10, 11], such as yield rate of memristor array and memristance variation, which degrades the performance of application program running on the memristor based computing system. In response to this, we propose a method to keep the performance of the binarized neural network running on memristor based computing system.

2 Binarized Convolutional Neural Network and Proposed Method

2.1 Binarized Convolutional Neural Network

The architecture of the binarized convolutional neural network used for simulation only two layers, which is proposed in our previous work [12]. And the detail information of the binarized convolutional neural network is shown in Fig. 1.

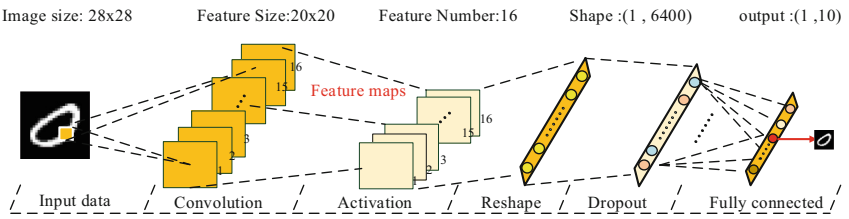


Fig. 1. Detail information of the binarized neural network

For the binarized convolutional neural network shown in Fig. 1, the input images of the network are first processed into binary, viz., the pixel value of them is processed to be 0 or 1. And the processing function is shown as follows:

$$f(x) = \begin{cases} 0\, x \leq 0.5 \\ 1\, x > 0.5 \end{cases} \tag{1}$$

The output type of the activation is the same as the input, viz., 0 or 1, and the express of the binarized function is shown as follows:

$$f(x) = \begin{cases} 0\, x \leq 0 \\ 1\, x > 0 \end{cases} \tag{2}$$

The binary form of the weight parameters in the binarized neural network is $+1$ or -1, and the processing function is shown as follows:

$$f(x) = \begin{cases} -1\, x \leq 0 \\ +1\, x > 0 \end{cases} \tag{3}$$

2.2 Proposed Method

The principle of the proposed method is to inject Gaussian noise into the binary weights and binary function of activation during the forward propagation of the training process. The purpose of injecting Gaussian noise into the weights is to improve robustness of the binarized neural network to device defects, while the counterpart of that is to improve the robustness of the network to neuron output variation.

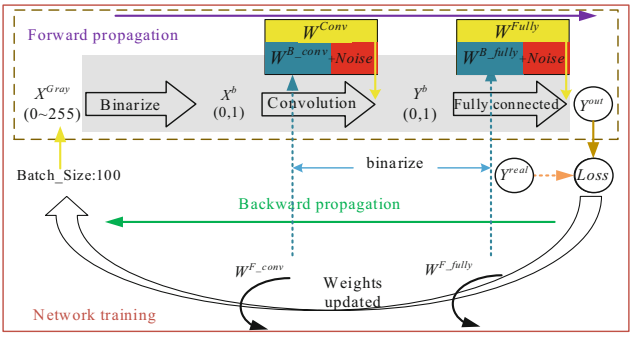


Fig. 2. Training process of binarized neural network

With Gaussian noise injected into the weights, the detail training process of the binarized convolutional neural network can be seen in Fig. 2. And it can be seen from Fig. 2 that the 'Noise' represents the random value sampled from Gaussian noise which follows normal distribution, and it is added to the binary weights, namely W^{B_conv} and W^{B_fully}, to get the weights W^{Conv} and W^{Fully}. Then, the weights W^{Conv} and W^{Fully} are used to perform convolution and vector-matrix multiply operation with inputs. In the weights updated phase of backward propagation, the weights W^{F_conv} and W^{F_fully} being float-point are updated according to the algorithm of gradient descent [13]. What should be noticed is that,since the gradient of the binary activation function at the non-zero point is 0, and the gradient at the zero point is infinite, we use the gradient of the tanh function to approximate the gradient of the binary activation function.

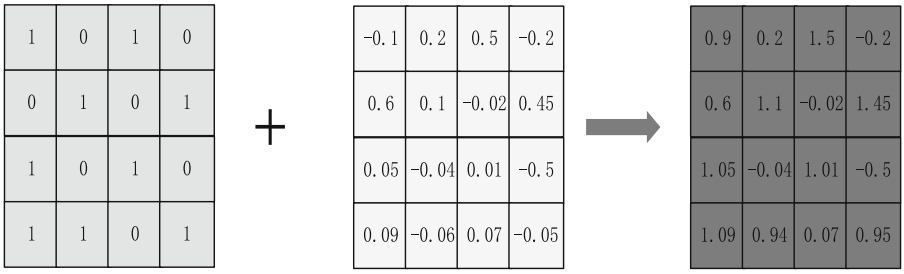


Fig. 3. Example of injecting noise into binary activation function

The implementation scheme of injecting the Gaussian noise into binary activation function can be seen in Fig. 3.

As can be seen in Fig. 3, the original outputs of the binary activation function only have two types of values, that is 0 and 1. And the value sampled from the Gaussian noise is float-point type. Therefore, the final type of the pixel value in the feature map is float-point.

3 Experiments

3.1 Simulation Settings

All the experiments in this study are conducted using a computer with 24 GB DDR4, Intel Core i7-8750H CPU (2.2 GHz), and a Nvidia GTX 1050 graphics card, and the Tensor flow [14] open-source library is used to train the binarized neural network. The simulation results are obtained using Monte-Carlo simulation method in Python. Another simulation settings are shown as following.

(1) Parameters of memristor model
During the simulation process, two Pt/HfO2:Cu/Cu memristors [15] are used for representing one weights in the binarized convolutional neural network. And the average resistance value of the memristors with high resistance state (HRS) or low resistance state (LRS) is 1 MΩ and 1 KΩ, respectively.
(2) Parameters for training binarized convolutional neural network
The MNIST dataset is divided into three subsets, viz., training set including 55,000 images, validation set containing 5000 images, and testing set composed of 10,000 images. The number of epoch for training network is 100, and the value of the batch size for gradient descent optimization algorithm is also 100. In addition to that, exponentially decaying learning rate is applied, and the initial learning rate is 0.01.
(3) Model of non-ideal characteristics
The defects considered in our experiments include three types, namely yield rate of the memristor array, resistance variation of the memristor and neuron output variation.

For the problem of the yield in memristor array, meaning that there are some damaged devices in the array and each damaged device either sticks at G_{HRS} (conductance value corresponding to memristor in the state HRS) or G_{LRS} (conductance value corresponding to memristor in the state of LRS), the resistance in memristor array is randomly changed to be G_{HRS} or G_{LRS} for emulating the yield rate problem. And an assumption has been made that there is 50% possibility for each damaged device being stuck at G_{HRS} or G_{LRS}.

As for the problem of the resistance variation of the memristor, the resistance of the memristors in the state of HRS (LRS) in array is not exactly 1MΩ (1KΩ), but fluctuates around 1MΩ (1KΩ). Therefore, during the simulation process, the model of the resistance variation is depicted as Eq. (4):

$$R\,N(\mu, \sigma_v^2) \tag{4}$$

In Eq. (4): the parameter μ represents the average value of the memristance in HRS or LRS, viz., 1MΩ in the state of HRS and 1KΩ in the state of LRS. The parameter σ_v should satisfy the relation described in Eq. (5):

$$\sigma_v = \mu \times r_v \tag{5}$$

In Eq. (5): the parameter r_v denotes to the scale of the resistance variation.

With respect to the problem of neuron output variation, meaning the logical value output by the binary activation function is not corresponded to the actual voltage value output by neuron circuit, the logical value +1 (0) is not exactly mapped to the output of the neuron, viz., + VCC (0V), but fluctuates around + VCC (0V). During the simulation process, the model of the neuron output variation is depicted as Eq. (6):

$$V\ N(\mu_1, \sigma^2) \tag{6}$$

In Eq. (6): the parameter μ_1 represents the expected voltage value of the neuron output, viz., + VCC and 0V, and the parameter σ is the standard deviation of the normal distribution reflecting the range of the neuron output variation.

3.2 Simulatino Results

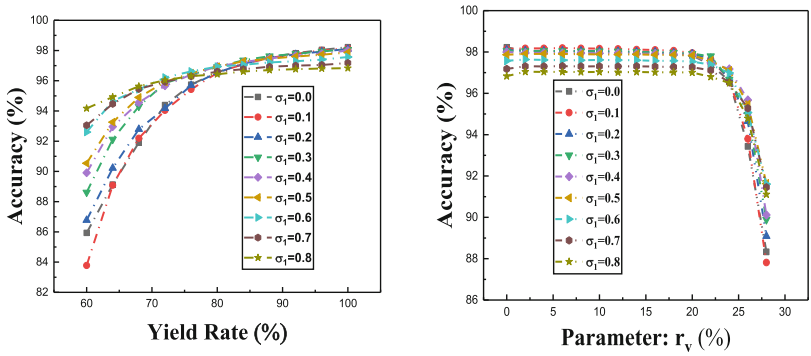
At first, the performance of the method with Gaussian noise injected into binary weights is first demonstrated. The robustness of the binarized convolutional neural network trained through the method with Gaussian noise injected into binary weights is analyzed based on the model of non-ideal characteristics.

Table 1 gives the information about the recognition rate of the network trained through method with noise injected into binary weights on MNSIT. What should be noticed is that, the parameter (σ_1) of the noise injected into binary weights is closely related to the parameter (σ_v) of the resistance variation model for the reason that two memristors forming a differential pair are used to represent one weight.

Table 1. The performance of the binarized convolutional neural network trained through method with noise injected into weights.

Gaussian noise injected into binary weights (σ_1)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Accuracy (%)	98.15	98.1	98.06	98	97.87	97.57	97.18	96.83

Figure 4 shows the analysis results of network's tolerance for yieldrate of the memristor array and resistance variation of memristor when the network is trained through or not through ($\sigma_1 = 0.0$) method of injecting noised into weights.What should be noticed is that, the noise parameter $\sigma_1 = 0.0$ means that the method of injecting noise into weight is not adopted.



(a) (b)

Fig. 4. Analysis results of the tolerance of binarized convolutional neural network for yield rate of the memristor array (a) and resistance variation of memristor (b).

As can be seen in Fig. 4, with the value of noise parameter σ_1 increasing, the network's robustness to yield rate of memristor array and resistance variation of memristor is improved, however, the performance of the network under ideal condition shows a gradual decline. Therefore, a reasonable noise parameter value should be given to balance the network performance and robustness. It can be noticed from table 1 that the recognition rate of the network achieves more than 97.5% when noise parameter varies from 0.1 to 0.6. And it can be seen from Fig. 4 (a) and (b), when the noise parameter is 0.6, the network not only has a good tolerance to the resistance variation of the memristor, but also has a good tolerance to the yield of the array. Therefore, the parameter value of the noise injected into weights is 0.6 in this paper. Figure 5 gives the analysis

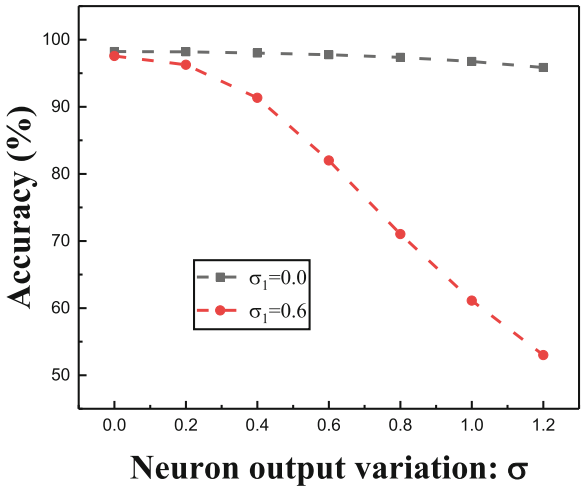


Fig. 5. Results of network's robustness to neuron variation.

results of the network's tolerance for neuron output variation when the noise parameter is 0.0 and 0.6, respectively.

What can be seen from Fig. 5 is that the network's tolerance to neuron output variation is degenerated. To improve the network's tolerance for neuron output variation, the method of injecting noise to binary activation function is also adopted during the training procedure of the network. Table 2 gives the information about the performance of the network trained with method of injecting noise into binary weights ($\sigma_1 = 0.6$) and binary activation function(σ_2).

Table 2. The performance of the network trained through method with Gaussian noise injected into weights ($\sigma_1 = 0.6$) and activation.

Gaussian noise injected into binary activation (σ_2)	0.2	0.4	0.6	0.8	1.0	1.2
Accuracy under ideal condition ($\sigma = 0.0$)	97.33%	97.13%	96.66%	96.55%	96.03%	95.66%
Accuracy when the parameter of neuron output variation ($\sigma = 1.2$)	67.99%	88.28%	91.44%	93.33%	93.62%	93.67%

What can be seen from Table 2 is that, as the noise parameter σ_1 is 0.6 and noise parameter σ_2 increase, the performance of the network under ideal condition declines continuously, but the tolerance of the network to neuron output variation increase gradually. Therefore, to keep the performance of the network excellent under ideal condition and improve the tolerance of the network to neuron output variation, we select a rough value for the noise parameter σ_2, that is 0.5. Similarity, the parameter (σ_2) is related to the parameter (σ) of the neuron output variation model for the reason that the neuron output variation follows normal distribution. Figure 6 shows the robustness of network trained through method with noise injected into weights ($\sigma_1 = 0.6$) and binary activation ($\sigma_2 = 0.5$) to non-ideal characteristics.

As can be seen in Fig. 6 (a) and (c), the robustness of the network trained through method with noise injected into binary weights ($\sigma_1 = 0.6$) and binary activation ($\sigma_2 = 0.5$) to yield of array and neuron output variation is improved. It also can be noticed from Fig. 6 (b) that the performance of the network under ideal condition declines marginally, which can be ignored.

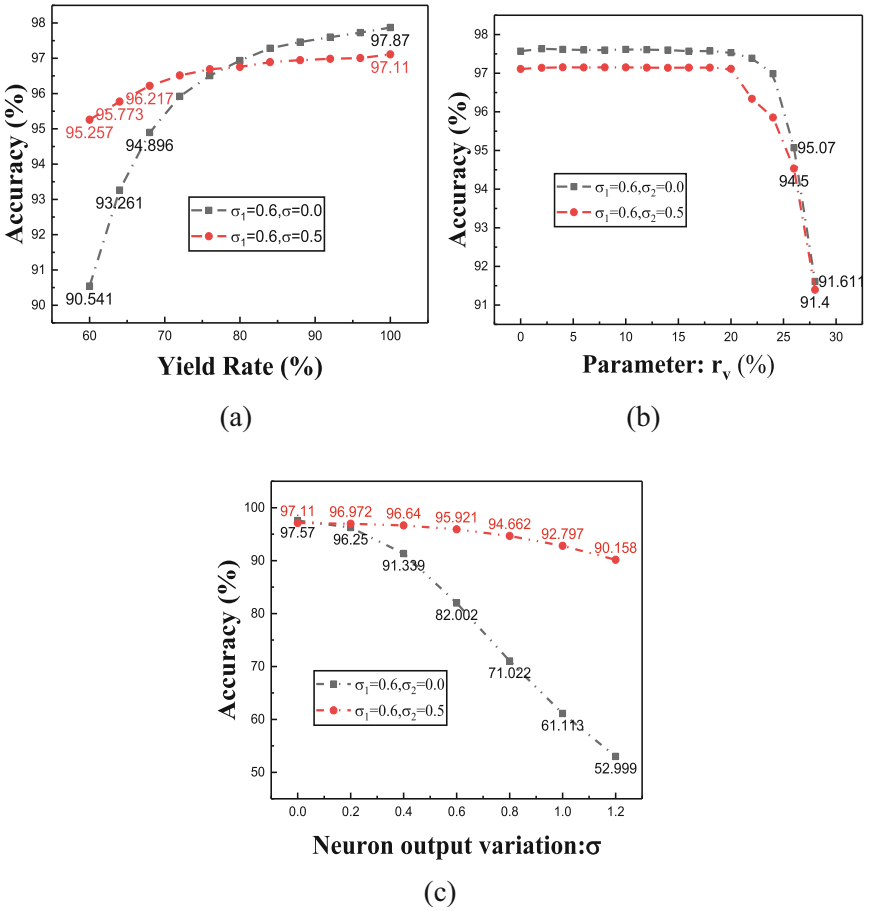


Fig. 6. The robustness of the binarized convolutional neural network trained trough method with noised injected into weights ($\sigma_1 = 0.6$) and binary activation ($\sigma_2 = 0.5$) to yield of array (a) and resistance variation of memristor (b) and neuron output variation (c).

4 Conclusion

In this paper, we propose a method for obtaining highly robust memristor based binarized convolutional neural network. By injecting Gaussian noise into binary weights and binary activation function during the training procedure, the reasonable noise parameter is selected for keeping the performance of the network and the network's tolerance to non-ideal characteristics. A binarized convolutional neural network is mapped into memristor array for simulation, and the results show that when the yield of the memristor array is 80%, the recognition rate of the memristor based binarized convolutional neural network is about 96.75%, and when the resistance variation of the memristor is 26%, it is around 94.53%, and when the neuron output variation is 0.8, it is about 94.66%.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61974164, 62074166, 61804181, 62004219, and 62004220).

References

1. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1 (2016)
2. Courbariaux, M., Bengio, Y., David, J.-P.: BinaryConnect: training deep neural networks with binary weights during propagations. Adv. Neural Inf. Process. Syst. **28** (2015)
3. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: imagenet classification using binary convolutional neural networks. Computer Vision - Eccv 2016, Pt Iv (2016)
4. Qiao, G.C., Hu, S.G., Chen, T.P., et al.: STBNN: Hardware-friendly spatio-temporal binary neural network with high pattern recognition accuracy. Neurocomputing **409**, 351–360 (2020)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Computer Science (2014)
7. Wulf, W.A., McKee, S.A.: Hitting the Memory Wall: Implications of the Obvious **23**(1), 20–24 (1995)
8. Strukov, D.B., Snider, G.S., Stewart, D.R., Williams, R.S.: The missing memristor found. Nature **453**(7191), 80–83 (2008)
9. Ielmini, D., Wong, H.: In-memory computing with resistive switching devices. Nature Electronics **1**(6), 333 (2018)
10. Kim, S., Kim, H.D., Choi, S.J.: Impact of synaptic device variations on classification accuracy in a binarized neural network. Sci. Rep. **9**(1), 15237 (2019)
11. Liu, B.Y., Li, H., Chen, Y.R., et al.: Vortex: variation-aware training for memristor x-bar. In: 2015 52nd Acm/Edac/Ieee Design Automation Conference; Los Alamitos (2015)
12. Huang, L., Diao, J., Nie, H., et al.: Memristor based binary convolutional neural network architecture with configurable neurons. Frontiers Neurosci. **15**, 328 (2021)
13. Lecun, Y., Bottou, L.: Gradient-Based Learning Applied to Document Recognition. 86(11), 2278-2324 (1998)
14. Abadi, M., Barham, P., Chen, J.M., et al.: TensorFlow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI); Nov 02–04, Savannah, GA (2016)
15. Liu, S., Wang, W., Li, Q., et al.: Highly improved resistive switching performances of the self-doped Pt/HfO2:Cu/Cu devices by atomic layer deposition. Science China-Physics Mechanics & Astronomy. 59(12) (2016)